Operating System Security: Adding to the Arsenal of Security Techniques[1]

Dave Ferraiolo, Peter Mell
David.ferraiolo@nist.gov, Peter.mell@nist.gov
National Institute of Standards and Technology
12-15-1999

Introduction

In recent bulletins and in the popular press, we have heard much about a variety of advanced or enhanced security mechanisms such as firewalls, intrusion detection, smart cards, public key infrastructure, digital signatures and many more. Indeed, many organizations are investing scarce resources on these mechanisms to secure their systems. Somewhat fewer organizations are spending time and money on risk management, policy development, incident handling, vulnerability analysis, security architecture and other vital activities. Nonetheless, many organizations still find themselves quite vulnerable to attacks on their computers and networks.

One of the principal reasons that organizations continue to have security problems is that application software often contains numerous vulnerabilities. Many security systems (such as firewalls, intrusion detection systems, and virus checkers) attempt to protect these insecure applications by monitoring and filtering the application's interactions with users. Each security product provides a different monitoring or filtering technique and the use of multiple techniques can form a strong barrier against attack. However, ultimately, all of these barrier techniques are inadequate because users must be allowed to interface directly with vulnerable software applications. If such software contains a previously unknown vulnerability, then most likely an attacker can exploit it without being stopped. Despite this, our best defense (apart from building secure applications) is to install ever-stronger barriers around our software. One of the best places for such a barrier is as close to an application as possible: the operating system (OS).

An OS has direct control over applications and can provide strong security services to, and around, an application. However, many OSs allow applications too much control and thus vulnerabilities in applications often lead to complete compromises of computers. OSs themselves often have flaws; nevertheless, much of the public continues to purchase OSs known to be insecure even when given the option for more secure systems. Some people knowingly buy insecure systems because they prefer convenience to security. As computer security incidents become more widespread and dangerous, this line of reasoning may quickly change.

The purpose of this bulletin is two fold. First, it provides an overview of some security features that have often been neglected in mainstream OSs. It describes the extent to which these features have been implemented and how users can take full advantage of the available capabilities. Second, it warns users that OS security along with most other mainstream security mechanisms is imperfect and can not stop all attacks. Despite this

[1] This paper was published as a National Institute of Standards and Technology, Information Technology Laboratory Bulletin, December 1999.

fact, using a combination of different security mechanisms can create a strong security barrier against attacks. Understanding the strengths and weaknesses of these techniques can aid one in the development of appropriate security policies, risk management plans, and in the purchasing of security technology.

Important OS Security Features

OSs, which directly control applications, can create a strong shell of security around inherently weak software. OSs can enhance security by providing secure communications between applications, limit penetrated applications from spreading their influence, and limit the leakage of critical information out of an application.

Several OS security mechanisms are described here to demonstrate the importance of OS security in protecting application software. These are trusted paths, least privilege, non-discretionary access protection, and tokens. We cite these mechanisms as illustrative examples of the importance of OS security not as a comprehensive or exhaustive list.

Trusted paths in OSs

A trusted path is a mechanism by which an entity (user, program, or hardware) may directly communicate with another entity on a host. This communication must have the properties that the communication cannot be intercepted by another entity and the two entities can mutually authenticate. In the absence of a trusted path mechanism, malicious software may impersonate “trusted” software to the user or may impersonate the user to the trusted function. As such, this feature defeats malicious software that would attempt to masquerade as other software or software that would secretly monitor keystrokes or inter-process messages.

Limited trusted path capabilities exist in some widely used OSs. Windows NT, for example, uses a trusted path mechanism to prevent Trojan horse programs from stealing logon passwords. When a user logs into a Windows NT system they should first press control-alt-delete even if a logon window is already present. That keyboard signal causes an exception handler to run that suspends all non-OS processes and then presents a window to the user asking for a username and password. Thus, assuming that the OS has not been compromised, the user has reasonable confidence that the logon window is owned by the OS and not by malicious code. Similarly, the user also has reasonable assurance that no application apart from the OS is monitoring the user’s keystrokes during logon.

While the Windows NT trusted path is useful, OS vendors are working on further expanding this capability to allow trusted paths between: programs, the OS, hardware devices (like smart cards), and users. The need for these expanded capabilities was highlighted by recently documented attacks in which a Trojan Horse applet captured credit card numbers, PIN numbers, and passwords by emulating a window system dialog box.

Implementing Least Privilege in OSs

On any computer system, certain programs must be granted the ability to bypass the security constraints normally imposed by the system. For example, to create a backup of all files, an administrator must run a program that is able to read all files even if the administrator is not normally allowed such access. Other programs must be given unusual system access, such as the programs to shut down the system, create new users, and repair damaged file systems. Historically, all of the programs that needed special privileges were run using a user ID called *root*, *superuser*, or *administrator*. This gave these programs that needed extra privileges complete control of a host, including the ability to bypass all security restrictions and limitations. This means that the backup program can be used to shut down the system. The shutdown program can create new users, and the program to create new user accounts can read all files on the system. The problem with this is that if any of these privileged programs has an exploitable bug, an attacker can use the program to do any action on the host.

To prevent this problem, some OSs provide mechanisms by which one can assign programs only the specific privileges that they need. This is done by breaking the root privilege into a set of smaller privileges and thereby limiting the programs to a relatively small set of privileges. By giving programs the least number of privileges needed, a privileged program that is penetrated by a hacker will not give the hacker complete control of the host. For example, a program designed to eject a CD-ROM from a host has only the privilege to control the CD-ROM drive and not the ability to read the password file. A hacker then could not, as was recently done on a major OS, penetrate the eject program and gain the ability to read the password file.

Many versions of Unix are especially susceptible to this problem since hackers can easily take advantage of the privileges of a penetrated program. When one program starts another, a newly created program runs with the user ID of the first program. This means that a malicious user who can exploit a bug in a root program may be able to start up an interactive root session. If a user is running as root, every program the user runs will have unlimited privileges on the system. The user can create any file, modify any file, and delete any file. The user can send and receive selected network packets, has the ability to intercept all packets on the network, and thus can view traffic between any two other hosts on the same network.

On many versions of Unix, the most a system administrator can do to implement the least privilege principle is to give as few programs as possible root status. The granularity of implementing least privilege in most version of Unix is to choose whether or not a process should run as root. Some programs must be run as root and there is no way around giving them complete control of the machine. However, many administrators mistakenly run programs as root that could be run under user accounts. System administrators should carefully evaluate the privileged programs on critical servers and determine which, if any, could be run with user privileges instead of as root. Ideally, standard Unix versions would give system administrators the ability to implement the least privilege principle with much finer granularity. Several vendors, however, do offer products that enhance the least privilege capabilities in Unix.

Window NT has a finer grained least privilege mechanism for processes and users. Every Windows NT process runs as some user identity. Every user identity in Windows NT can be given a set of rights on that host. There exist at least 34 rights that can be given or denied to each user account. Example rights are: changing the system time, managing security logs, accessing a computer over the network, taking ownership of files, and creating users. Despite this advanced least privilege mechanism, all but two services (OS processes and programs that run in the background) included with Windows NT must be run as the system user which gives them complete control of the host. To fully take advantage of the least privilege mechanism available in Windows NT, system administrators should create a separate user account for each service and give it only the privileges needed to run that service.

The use of a strong least privilege mechanism can eliminate many of the most commonly reported security problems with standard OSs, including many buffer overflow attacks. Even if a specific privileged program bug goes unfixed, use of the least privilege principle prevents the bug from allowing a malicious user to bypass system security completely.

Non-discretionary Protection in OSs

Most OSs provide what is called discretionary access control. This allows the owner of each file, not the system administrator, to control who can read, write and execute that particular file. Another access control option is called non-discretionary access control. Non-discretionary access control differs from discretionary access control in that the definition of the access rules are tightly controlled by a security administrator rather than by ordinary users.

Non-discretionary access control can help ensure that system security features are enforced and tamperproof. With this technique, security administrators can ensure that critical files are properly write-protected and viewable by only a trusted set of people. Non-discretionary mechanisms can also be used to protect against inadvertent execution of untrustworthy applications since users can execute only those programs that they are allowed to execute. With discretionary access control, a user may have carefully defined a file protection policy but a virus could change that policy making the user's files open to the world. This scenario is not possible in a non-discretionary access control environment.

Non-discretionary access control provides an organization with tighter security than is available with discretionary access control. However, this comes at a cost. Additionally, users may resist having file control policies specified by upper management. Also, it is time consuming to manage what groups of users should have access to what files (although sophisticated software exists to make this easier.) Despite these drawbacks, organizations requiring a high level of security as well as organizations that cannot depend upon users' voluntary adherence to site security policy should consider non-discretionary access control mechanisms. Non-discretionary access control capability can be added to many OSs with add on software.

Integration of OS with Security Tokens

OSs in the near future will tightly integrate with a variety of small computing devices. These devices may take the form of electronically enhanced cards, rings, or a variety of other wearable objects. Smart cards (credit cards with embedded computer chips) are an example of such devices and they are widely used in Europe. These devices can perform cryptographic services to verify a person's identity, digitally sign transactions, or scramble information to be readable by only the owner of the device. OSs must tightly integrate with these devices so that rogue programs on a host can not deceive the devices into performing unauthorized encryption services. Only those applications authorized by a user should be able to perform transactions with the encryption device. To provide this type of security, future OSs must be made aware of these devices and provide trusted paths between the device and applications only when authorized by the user.

The Weaknesses of Barrier Security Technologies

Past *ITL Bulletins* have recommended security techniques that can help stop the majority of computer attacks. Some recent bulletins of this type include:

- "Acquiring and Deploying Intrusion Detection Systems" describes how intrusion detection systems can detect attacks upon a network (November 1999)
- "Securing Web Servers" focuses on specialized issues and techniques for securing web servers (September 1999)
- "Computer Attacks: What They Are and How to Defend Against Them" provides general solutions for protecting a network (May 1999)

These bulletins are available on the web at: http://www.nist.gov/itl/lab/bulletns/cslbull1.htm.

However, no combination of these security barrier techniques is sufficient to guarantee resistance to determined attacks. This includes the use of OS security techniques. Each technique has a weakness that can be mitigated by the use of multiple barriers, but which ultimately will not be attack proof. Since users must interface directly with insecure software, security barriers are unable to stop attackers from exercising all application software flaws. In practice, however, a combination of these techniques provides a reasonably strong barrier against most attackers.

Using Patches to Enhance Security

One of the most common methods for plugging known security flaws is the installation of the latest vendor supplied security patches. Patches are programs that fix errors in software. However, patching systems is not a perfect security solution. First, the constant stream of patches can quickly overwhelm administrators who are already burdened with other administrative tasks. Second, even though organizations install all of the latest patches, new attacks via the Internet will continue. When new attacks are discovered and published on the Internet, a large number of networks will become instantly vulnerable to attack until new patches are created and installed. Several weeks or months may elapse before an effective patch can be prepared to counter a new attack, leaving affected servers wide open to attack. Organizations can maintain their awareness about new patches by monitoring security advisories about threatening or popular attacks. These advisories are issued by a variety of organizations, and usually reference a patch or work-

around that will fix the discussed vulnerability. The most popular source of security advisories comes from the Carnegie Mellon Emergency Response Team at www.cert.org. In addition, we suggest you consult with www.fedcirc.gov.

Firewalls

Firewalls police network traffic that enters and leaves a network. A firewall may completely disallow some traffic or may perform some sort of verification on other traffic. These features enable well-configured firewalls to stop a large number of publicly available attacks. For example, firewalls can stop many TCP based denial of service attacks by analyzing TCP packets and throwing away those that are maliciously formed. Firewalls can stop many penetration attacks by disallowing many protocols that an attacker could use to penetrate a network. By limiting access to host systems and services, firewalls provide a necessary line of perimeter defense against attack. However, firewalls do not in most environments adequately reduce the risk for applications that generate active content or implement transaction-oriented services. For example, firewalls do not typically have the processing power or ability to analyze downloaded Java™ applets. As the term implies, a firewall restricts overall access from an untrusted environment (the Internet) to a friendly environment (the local company network). The new paradigm of transaction-based Internet services makes these "perimeter" defenses less effective as the boundaries between friendly and unfriendly environments blur. A firewall controls broad access to all networks and resources that lie "inside" it. Once packets from a user have traversed the firewall and been authorized to enter the internal network, the firewall cannot prevent access to or modification of specific resources—in the worst case, the system security data itself. For Internet-based transaction systems, the security mechanisms must be able to provide or deny access to particular web pages, applications, and databases on the basis of individual user profiles or server authentication. Firewalls are unable to provide such detailed security measures, important as they are to total systems security solutions.

Virus Detection Software

Virus checkers monitor computers and look for malicious code. Virus checker software must be installed on all computers that are to be monitored and should be updated frequently for maximum effectiveness. Virus checkers on e-mail servers that scan e-mail attachments should supplement virus checkers on hosts. This way, the majority of viruses can be stopped before they reach the users. However, virus detection software can only detect viruses that a vendor has analyzed and programmed into the software. Viruses that are custom built by attackers for a particular organization or person will escape detection. In addition, fast spreading viruses can infect large portions of the Internet before virus detection manufacturers can release software updates that fight the new threat.

Intrusion Detection

Intrusion detection is the process of detecting unauthorized use of, or attack upon, a computer or network. Intrusion detection systems (IDSs) are software or hardware systems that detect such misuse. IDSs are effective tools that should be employed by any large organization; however, they are not standalone security mechanisms. IDSs, for the most part, detect attacks that have occurred, but they usually cannot prevent attacks.

Furthermore, they normally are only able to detect attacks that have been previously seen and analyzed by the IDS vendor. Thus, novel or recently published attacks can be launched undetected against a network using IDSs. Some IDSs can launch limited responses to detected attacks, but these responses are usually not sufficient to stop sophisticated attackers. IDSs are strong security mechanisms only when used in conjunction with a variety of preventative security techniques.

Encryption

Some believe that when encryption becomes widely used, especially for network connections, that there will be no need for other security techniques. Unfortunately, encryption does not provide a complete security solution. Encryption can protect data in transit and can protect data stored on a server, but attackers may still obtain the data in several ways. Whenever the encrypted data is used, it must be first decrypted. A clever attacker can simply copy the decrypted information as it is decrypted. This can be accomplished by replacing the decryption program with a version that allows the attacker to copy the decrypted data. Another technique is to steal the encryption keys. These keys may reside unprotected on a host or an attacker might have to monitor a user's keystrokes to discover these passwords. Thus, data protected by a "strong" encryption system that would take a supercomputer 100 years to crack can be recovered in seconds by an attacker that steals the encryption keys.

Vulnerability Scanners

Vulnerability scanners are programs that scan a network or hosts looking for computers that are vulnerable to attacks. Scanners use a large database of identified vulnerabilities to probe computers in order to determine the vulnerable ones. Both commercial and free vulnerability scanners are available. They are very effective at finding vulnerable hosts, but they can only look for previously identified vulnerabilities. Newly released attacks and attacks that are not publicly known will not be revealed by the use of vulnerability scanners.

Evaluations

Most security managers, already hard-pressed to maintain daily systems operations, face significant barriers to incorporating new technologies and adequate systems security. They depend almost exclusively on vendor information about security performance when they install new software or upgrade existing security software on their systems. Given the potential implications of security system failure, it is critical that managers seek out security solutions that have undergone independent evaluation, testing, and validation. One of the largest evaluation efforts is the National Information Assurance Partnership (NIAP) run jointly by NIST and the NSA. See http://niap.nist.gov for details on evaluated products.

While the use of evaluated software is a necessary step in the direction of improved security, it does not guarantee the security of an organization. Product evaluations can find flaws and increase the level of trust in a product, but they do not guarantee the absence of all flaws. Even when using evaluated products, organizations must implement many other security mechanisms to create an in-depth defense strategy.

Conclusion

The best known method of securing a network or host is to use multiple security technologies together as part of a well thought out security plan. Each security technology has a weakness, but together security devices can create strong barriers against attacks. Outside audits of such security plans can be beneficial by highlighting weak points. Be aware though, that in general, no combination of security technologies can completely secure an organization and one must be prepared to respond to successful attacks.

OS security technologies necessarily fit into any security plan because all of the applications one is trying to defend run on top of OSs. One should ensure that the security capabilities of an organization's OSs are fully utilized. Furthermore, one should plan future OS purchases based on the security of the OS itself as well as the security features it provides. More security conscious organizations should consider purchasing add on software that enhances the security of their OSs.

NOTE: Any mention of commercial products is for information only; it does not imply recommendation or endorsement by the National Institute of Standards and Technology nor does it imply that the products mentioned are necessarily the best available for the purpose.